APPLIED COMPUTATIONAL FLUID DYNAMICS
Chapter 5
Inlet, Duct, and Nozzle Flows

Charles E. Towne

NASA Lewis Research Center
Cleveland, Ohio

Contents

Principal Notation

a, b	Semi-axes in *Spirit* inlet cross section
DLP(core)	Core compressor distortion limit parameter
DLP(fan)	Fan tip distortion limit parameter
h	Vortex generator height
M_{inf}, M_{thr}	Free stream and throat Mach numbers for *Spirit* inlet

p	Static pressure
Pr	Prandtl number
R	Equivalent throat radius in *Spirit* inlet
Re	Reynolds number
Re_R	Reynolds number based on throat conditions for *Spirit* inlet
t	Streamwise marching parameter in *RNS3D* code
T_{exp}, T_{ext}, T_{sys}	Measured thrust, thrust due to external hardware, and true system thrust
$w\sqrt{\theta}/\delta$	Corrected engine airflow
x_{CL}, y_{CL}	Centerline coordinates for *Spirit* inlet
δ	Boundary layer thickness

Subscripts

nf	Value at no-fuel-flow condition
r	Reference quantity
w	Wall value

5.1 Introduction

The use of computational fluid dynamics (CFD) for the design and analysis of flow through inlets, ducts, and nozzles has increased greatly in the last few years. Several factors have contributed to this growth, including: (1) complex geometric design requirements, leading to flow phenomena that are outside our established base of experience and may not be intuitively predictable; (2) high fuel costs, leading to potentially large cost savings for even small performance improvements; (3) the high cost and/or lack of facilities for extensive experimental testing; (4) the continued development and improvement of sophisticated numerical algorithms for solving the complex equations governing fluid flow; and (5) the tremendous improvements in computational power.

It's interesting to note, though, that some of the analysis methods used in CFD pre-date the computer. In 1929 Prandtl and Busemann applied the method of characteristics using a graphical technique to design a two-dimensional supersonic nozzle. A few years later they designed a nozzle for the first practical supersonic wind tunnel (Anderson, 1982). Not surprisingly, though, the real growth of CFD as a practical way of solving real-world problems began with the introduction of the digital computer in about the mid-1960s.

Today CFD is being applied by industry, government, and universities to duct flow problems in a wide variety of areas. The principal users are probably in the aeronautics and space areas, for flows in a variety of ducts, such as jet engine inlets and nozzles, fuel lines and storage tanks, wind tunnels, and rocket nozzles. It's also being used in many other areas, including: in the automotive industry to design air ducts, cooling lines, and hydraulic lines; in the heating and air conditioning industries to design air ducts and to study room air flows; and in medicine to study blood flow.

Although CFD is also being widely used for compressors, turbines, and combustors, the focus in this chapter is on non-rotating and non-reacting flows. It should also be noted that the author's area of interest is inlets, ducts, and nozzles for airbreathing propulsion systems, and some of that bias will no doubt be apparent in the following discussion.

In this chapter we will first describe, in general terms, the steps involved in applying CFD to an inlet, duct, or nozzle problem. Two examples will then be presented showing how CFD is being applied to real problems in the aerospace field today. The first deals with using CFD to help design a subsonic inlet for a new general aviation jet aircraft, with the focus on lowering the total pressure distortion levels at the duct exit. This example is described in quite a bit of detail to clearly illustrate the steps involved in a real-world

application. In the second example, CFD is being used as a critical tool in the reduction of data from a hypersonic engine experiment, allowing thrust data to be obtained that couldn't be obtained in any other practical way. We will conclude with a discussion of the current status of CFD for inlet, duct, and nozzle applications, and what needs to be done if CFD is to become more widely used for routine real-world design work.

5.2 CFD Solution Process

5.2.1 Gather Information/Choose Analysis Method and Flow Models

The first step in any CFD study is to gather the necessary information about the problem. The amount of detail required will depend to some extent on how the results will be used. For example, a calculation being done to quantify the various sources of total pressure loss in an air duct will likely require a high level of fidelity in simulating the problem, with a detailed description of the actual geometry, incoming flow profiles, boundary conditions, etc. Conversely, if a parametric study is being done to determine which of several ducts has the lowest losses, the absolute loss level may not be critical, and a lower fidelity simulation may be acceptable.

The information gathered in this step will be used to determine: (1) the type of CFD analysis and flow modeling (e.g., turbulence modeling) that's appropriate for the problem; (2) the particular code to be used, based on how well its features and capabilities match those needed to solve the problem; and (3) the input parameters that will be used when running the code.

Some knowledge, or at least an experience-based guess, about the type of physical phenomena expected in the flow is required at this stage. The more that is known about the flow, the better the CFD simulation is likely to be. Some of the things that should be considered at this point are:

- *Will the flow be compressible or incompressible?*

  Depending on the application, compressibility effects may become important for Mach numbers above 0.3 or so. A CFD code designed to compute incompressible flow will be useless for a compressible problem. A compressible code may be used at low Mach numbers, but iterative methods may take longer, possibly much longer, to converge. Some compressible codes, however, use numerical techniques such as matrix pre-conditioning that allow them to be used at "incompressible" conditions without suffering from slower convergence rates.

- *Will the boundary layers be thin or thick?*

  If viscous effects can be neglected, Euler or even potential flow methods are probably the ones to use. They should be faster, both because the viscous terms do not have to be computed, and because fewer grid points are needed since there are no shear layers to resolve. If viscous effects are important, but the boundary layers are thin, a boundary layer method may be used in conjunction with an inviscid method to compute these effects. If the boundary layers are thick, a fully viscous method such as a parabolized or Reynolds-averaged Navier-Stokes analysis will be necessary.

- *Are regions of separated flow possible?*

  Some parabolized codes include approximations that allow them to compute flows with small separation bubbles, but larger regions of flow separation can only be computed by solving the Navier-Stokes equations.

- *Will the flow be laminar, turbulent, or transitional?*

  If the flow is turbulent or transitional, a turbulence model must be used. There are a wide variety of turbulence models in use today, and most CFD codes for viscous flow have more than one to choose

from. In most cases, the choice will be between an algebraic model, a one-equation model, and a two-equation model.[1] Unfortunately, there is no single turbulence model that works well for all types of flow, and experience is invaluable when choosing an appropriate model for a particular problem. As one might expect, algebraic, or zero-equation, models are the simplest and fastest, and are most suitable for relatively simple attached flows. Two-equation models are more complicated and slower, but are more suitable for complex flows with shear layer interactions and/or flow separation. One-equation models fall somewhere in between. Some, but by no means all, of the models may be able to to predict laminar-turbulent transition, but this capability is even less mature than the prediction of fully-turbulent flows. Turbulence modeling is currently a very active research area, and will probably remain so for at least several more years.

- *Will shock waves be present?*

  Shock waves are always a possibility in transonic or supersonic flow. Some CFD codes for inviscid flows do shock fitting, in which the shock waves are computed separately and "fitted" into the solution as infinitely thin discontinuities. Most codes, though, do shock capturing, in which the shocks are automatically "captured" during the solution of the governing flow equations. With these codes, the discontinuity is smeared, typically across 3–5 grid points.

- *Will three-dimensional effects be important?*

  All real flows are three-dimensional, but if the three-dimensional effects can be neglected, using a 2-D code will be significantly faster than using a 3-D code. Be aware though, that three-dimensionality may be important in a nominally "2-D" flow. One example is flow through a rectangular cross-sectioned supersonic inlet, where the sidewall boundary layers, and their interaction with the shock waves from the ramp, can play a critical role in determining the distribution of flow in the inlet.

- *Will real-gas effects be significant?*

  Most CFD codes assume that the fluid is thermally perfect, i.e., that it satisfies the thermal equation of state

  $$p = \rho R T$$

  Some codes also assume a calorically perfect gas, i.e., one for which the ratio of specific heats $\gamma = c_p/c_v$ is constant. The molecular viscosity and thermal conductivity laws that are needed should also be determined, at least for laminar flows. (For turbulent flows, the turbulent values will likely overwhelm the molecular values, except very near solid surfaces where the turbulent values approach zero.) For the molecular viscosity coefficient, most CFD codes use either a power-law approximation, such as

  $$\frac{\mu}{\mu_r} = \left(\frac{T}{T_r}\right)^{0.67}$$

  which for air is valid at temperatures between about 300 and 900 °R (167 and 500 K), or Sutherland's Law

  $$\mu = C_1 \frac{T^3/2}{T + C_2}$$

  where C_1 and C_2 are constants, which is valid between about 180 and 3400 °R (100 and 1889 K) (Ames Research Staff, 1953). The thermal conductivity coefficient may be computed using similar equations, or related to μ through the Prandtl number $Pr = c_p \mu / k$.

  At hypersonic Mach numbers, however, temperatures may be high enough that the assumption of a perfect gas is no longer valid, and real-gas effects become important. Under these conditions, the

[1] The terminology "one-equation" and "two-equation" refers to the number of differential equations that are solved in the model. There are even more complex models, called Reynolds stress models, that are being used, but in general these are not yet practical for most engineering applications. There are also codes that actually compute, as opposed to model, the turbulent eddies. These codes are *very* long-running, even for simple configurations, and this is still very much a research area.

specific heats are not constant, and the perfect gas equation of state no longer holds. An equilibrium air model is sometimes used, which generally consists of an empirical curve fit or table of gas properties as a function of pressure and temperature. For hypersonic nozzle flows where chemical reactions may occur, a non-equilibrium finite-rate chemistry model may be required (Numbers, 1994).

- *What are the appropriate reference conditions?*

  While the mechanics of this will vary from code to code, reference conditions must be specified to define the state of the flow. Three critical parameters are the Mach number M_r, the Reynolds number $Re_r = \rho_r u_r L_r/\mu_r$, and the Prandtl number $Pr_r = (c_p)_r \mu_r/k_r$. For a duct flow, these parameters are typically based on values at the duct entrance, or at some critical location like the throat in an inlet. The Prandtl number is a function of temperature, but is approximately constant for most gases, and is often assumed to be constant in CFD codes.

- *What are the appropriate boundary conditions?*

  The specification of boundary conditions is very important. After all, since the equations governing fluid flow are the same for every problem (i.e., the Navier-Stokes equations), the boundary conditions are really what determine the solution. Again, the mechanics of specifying the boundary conditions will vary from code to code. All methods, however, require information at the inflow boundary, which may range from a complete description of the flow profiles, to just a specification of the mass flow rate. Euler and Navier-Stokes codes also require information at the outflow boundary, such as the static pressure or mass flow rate. At solid wall boundaries, inviscid methods require the velocity to be tangent to the boundary, and viscous methods generally use the no-slip (i.e., zero velocity) condition. Either the wall temperature or heat transfer rate will also usually be required. Specialized boundary conditions may also be needed, such as the capability to specify bleed flow rates.

The analysis method used to solve a problem will, ideally, depend on the information described above. For most internal flow problems the choice today is between an Euler, parabolized Navier-Stokes (PNS), or Reynolds-averaged Navier-Stokes method, although potential flow methods have also been used for some applications. In practice, the choice of a specific code, and to some extent the analysis method and flow models to be used, is also determined by the computer resources available, the availability of the code, the level of user expertise, and the level of confidence in the code within the organization.

5.2.2 Define the Geometry

The geometry may be initially provided in a variety of ways. In many modern designs, the geometry is the output from a Computer Aided Design (CAD) system. For relatively simple configurations, it may also be defined analytically. Another option is simply to define the surfaces by a series of points in some coordinate system. In any case, it must be defined in sufficient detail to allow the CFD simulation to meet the goals of the study.

There is often a compromise that must be made at this point, though, as was alluded to earlier. An exact representation of the real geometry would potentially yield the best solution, but may require a more sophisticated CFD method than would otherwise be needed, and use prohibitively large amounts of computer resources. As an extreme example, in a curved cooling duct there may be rivet heads or other protuberances that locally affect the flow. Including these in the CFD calculation may yield a very good simulation of the real-world flow, but would likely require a Navier-Stokes analysis and a very dense grid to resolve the flow details. But if the objective of the calculation is only to compare the flow profiles at the exit for two different turning angles, small details like this should not be included, as they are unlikely to affect the results.

Sometimes the effects of small geometric features like this may be modeled, instead of actually computed. One example might be including the effect of rivet heads on the flow, without actually including them in

the CAD geometry description, by specifying a surface roughness factor in the turbulence model. Another example is described as part of the Paragon *Spirit* inlet case in Section 5.3, in which the effects of vortex generators on the flow are modeled.

5.2.3 Generate the Grid

The next step is to generate the computational grid. To at least some extent, the exact type of grid that is used will be dictated by the choice of CFD code used to compute the flow. For internal flows, body-fitted grids are generally used, which map the irregularly-shaped physical flow domain into a rectangular computational domain. The surfaces of the geometry, and the inflow and outflow boundaries, become boundaries in the computational domain, which greatly simplifies the application of the numerical boundary conditions.

If the CFD code has multi-block capability (and most modern ones do), it may be desirable to divide the flow domain into blocks. A grid is generated for each block, and each block is solved separately by the CFD code, with interface boundary conditions used to pass information between blocks. This can greatly simplify the grid generation process for complex geometries. It may also be more efficient in the flow solution step by allowing parallel execution, with all the grid blocks being solved simultaneously on separate processors.

Grid generation usually proceeds by first constructing grids on the computational boundaries, then filling the interior of the flow field. For relatively simple geometries it may be possible or even desirable to write a customized program to generate the grid algebraically. This allows a great deal of user control over the distribution of grid points, and is especially useful when the flow will be computed several times with systematic variations in the grid. More complex geometries will require the use of a generalized grid generation program. These typically generate the grid by solving a set of elliptic partial differential equations. The CFD analyst doesn't necessarily need to be an expert in the inner workings of the grid generation program, but it helps to be proficient in using the program. For complex geometries, the grid generation step may well be the most time-consuming one in the entire solution process.

There are several points to keep in mind when generating the grid for an application. First, grid points should be concentrated in regions where large flow gradients are expected, to adequately resolve the flow there. These high-gradient regions include boundary layers near solid surfaces, shear layers between adjacent streams or in wakes or jets, and shock waves. In addition, abrupt changes in grid spacing can lead to numerical problems, so the change in grid spacing should be smooth. The distribution of grid points may be controlled in a variety of ways, depending on the particular grid generation program being used. In the future, adaptive gridding techniques may allow this to be done automatically as the flow solution is being computed, but current CFD production codes do not generally have this capability.

While the grid must be dense enough to allow an accurate solution, using too many grid points is wasteful of computer resources. Determining the degree of resolution necessary is somewhat of an art, learned through experience with a particular code and type of flow. In general, though, for turbulent boundary layer calculations the grid point adjacent to the wall should have a y^+ value below 1.0.[2] As noted earlier, shock waves in most CFD codes are captured, and smeared across 3–5 grid points. Ideally, then, the grid should be dense enough around the shock that this amount of smearing is acceptable.

Second, highly nonorthogonal grids should be avoided if possible. Even though many CFD codes solve the governing equations in generalized nonorthogonal coordinates, excessive grid skewness (i.e., nonorthogonality), especially near boundaries, may adversely affect the solution. For example, if "normal-gradient" boundary conditions have been implemented by simply setting the value at the wall equal to the value at the adjacent interior grid point, an error will be introduced if the grid is nonorthogonal at the boundary.

And third, the grid should be smooth. Sharp changes in slope and/or curvature of the grid lines will

[2] y^+ is the inner region coordinate in the boundary layer law of the wall, given by $y^+ = u_\tau y/\nu_w$, where the friction velocity $u_\tau = (\nu \partial u/\partial y)_w^{1/2}$ and y is the distance from the wall.

cause sharp changes in the metrics of the transformation between physical and computational space, and can lead to numerical difficulties.

Traditionally, CFD codes have used structured grids, which consist of curvilinear sets of points whose coordinates are specified by 3-D arrays in the three spatial directions. The discussion in this section, at least in part, has assumed that a structured grid would be used. However, some newer CFD codes use unstructured grids, which usually consist of triangular or tetrahedral cells whose coordinates are specified on a point-by-point basis. In general, unstructured grids may be created more quickly than structured grids, especially for complex geometries. To date, they have been used mostly with Euler solutions, and some question their suitability for viscous flows with thin shear layers. This is an active research area, though, and the use of unstructured grids in CFD is likely to become more widespread.

5.2.4 Compute the Flow Field

For configurations and flow conditions that are within the user's established experience base for the code being used, the CFD calculation itself is often the easiest step in the process. For the most part, it's a matter of setting up the necessary input file(s), specifying the necessary job control information for the computer system being used (e.g., linking files to the proper Fortran I/O units), and starting the job interactively or submitting it to a batch queue. Long-running calculations are usually done in steps, with each successive computer run restarting the calculation where the previous one left off. For iterative methods, the results should be examined after each computer run to check the convergence status, and to identify physically unrealistic results that indicate a problem with the program input or the computational mesh. See the following section for more information about analyzing the computed results.

For configurations and/or flow conditions that are outside the established experience base, however, the CFD calculation may take significantly longer. Most codes have a variety of input options for things like: the choice of artificial viscosity model, and the magnitude of artificial viscosity; the solution algorithm and iteration control parameters; the form of the boundary conditions to be used; and the choice of turbulence model, and all of its various input parameters. The proper values to use for a new case are often not obvious. The mesh density and quality that is required may also not be known. Under these conditions, it may take several "iterations" — running a case, examining the results, changing the input and/or mesh, and re-running the case — to get the first good result for a new case. Unfortunately, successfully running a CFD code is, at least for some applications, a combination of art and science.

5.2.5 Analyze the Results

After each intermediate computer run, as noted above, the results should be examined to check the convergence status, as well as to identify physically unrealistic results that indicate problems. One way to check convergence is to examine the L_2 norm of the residual for each equation. Ideally, the residuals would all approach zero at convergence. In practice, however, for real-world problems they often drop a certain amount and then level off. Continuing the calculation beyond this point will not improve the results. A decrease in the L_2 norm of the residual of three orders of magnitude is sometimes considered sufficient. Convergence, however, is in the eye of the beholder. The amount of decrease in the residual necessary for convergence will vary from problem to problem, and will depend on how the computed results are to be used. For some problems, it may be more appropriate to measure convergence by some flow-related parameter, such as the pressure or skin friction distribution along a surface. Determining when a solution is sufficiently converged is, in some respects, a skill best acquired through experience.

CFD codes are capable of generating a tremendous amount of flow field data. While some meaningful output may be created by the CFD code itself, examining the computed results generally requires the use of some sort of post-processing routine. A post-processing routine manipulates the results generated by the

CFD code, and presents them to the user in a form that is meaningful for the problem being studied. In general, this means looking at the results graphically.

There are a variety of graphics packages available, both from commercial vendors and in the public domain, that may be used to display the results from a CFD calculation. These range from simple 2-D x-y plotting programs to fully 3-D interactive graphics systems. With the 3-D systems available today, results may be displayed and examined in almost any form imaginable. Pressures may be plotted on the surface of a diffuser, for example, in the form of colored contour lines, or as filled and shaded polygons. Velocity vectors may be displayed showing the development of secondary flow vortices in a curved duct. Animations may be created showing unsteady phenomena, or tracking streamlines through a flow field. These interactive post-processing systems can be tremendously useful in identifying problem areas, and in understanding the critical physics of the flow.

Generalized post-processors like these may not compute all the parameters of interest for a specific application, however. Things like the compressor-face distortion values in an inlet, for example, or the integrated thrust in a nozzle, are not usually computed by general-purpose CFD codes or post-processing packages. It is sometimes necessary, therefore, to write a special-purpose code, or locate one that someone else has written, that reads the output files created by the CFD code and computes the needed values. When writing a post-processing code like this, it's often tempting to do a "quick and dirty" job, that works for the specific problem at hand but no others. For all but the most unique situations, however, it will pay off in the long run if it's written in as general a form as possible, to facilitate its application to other problems.

5.3 Example — Paragon *Spirit* Inlet

In 1995 Paragon Aircraft Corporation asked NASA Lewis Research Center for help in the design of the subsonic inlet for the *Spirit*, a new six-passenger general aviation jet aircraft they were developing. The inlet, shown in Figure 5.1, has an S-shaped centerline and a slightly elliptical cross section that transitions to a circle at the compressor face. The area ratio is 1.14.

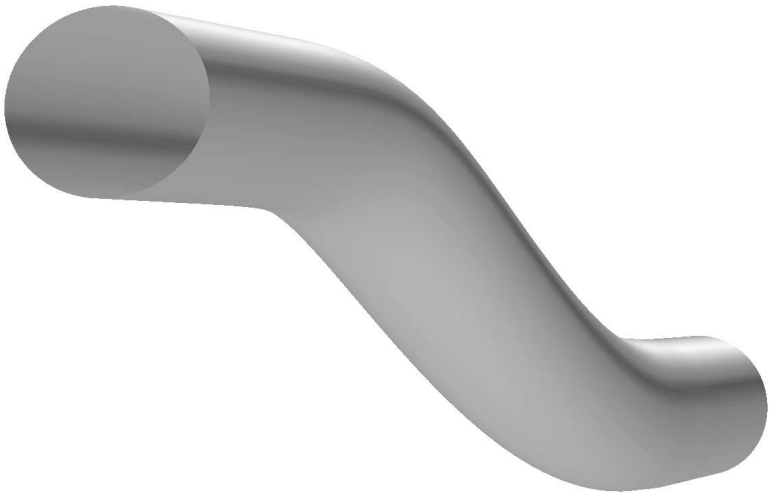


Figure 5.1: Paragon *Spirit* inlet

The objective of the study was to determine the baseline performance of the inlet, and to design a vortex generator system for the inlet that would ensure that it would meet the distortion criteria for the aircraft's

medium bypass ratio commercial turbofan engine.[3] In particular, values for total pressure recovery and distortion at the exit of the inlet were required.

5.3.1 Gather Information

Three operating points were of interest, as defined in Table 5.1.

Table 5.1: Operating Points for *Spirit* Inlet Calculations

Operating Point	**Altitude, ft (m)**	M_∞	$M_{\text{textitthr}}$	$w\sqrt{\theta}/\delta$, **lb$_\mathbf{m}$/sec (kg/sec)**	Re_R
High-speed cruise	36,000 (10,973)	0.70	0.660	75.3 (34.1)	1.14×10^6
Best cruise	40,000 (12,192)	0.50	0.542	66.4 (30.1)	0.74×10^6
Takeoff	Sea level	0.20	0.468	59.4 (26.9)	2.25×10^6

In the table, M_∞ and $M_{\text{textitthr}}$ are the free stream and throat Mach numbers, respectively; $w\sqrt{\theta}/\delta$ is the corrected engine airflow, where w is the actual engine airflow, θ is the ratio of the engine face average total pressure to standard sea level pressure, and δ is the ratio of the free stream total temperature to standard sea level temperature; and Re_R is the Reynolds number based on throat conditions and the equivalent throat radius (i.e., the radius of a circle with the same area as the elliptical throat).

It was known that the flow in the inlet would be subsonic but compressible, with turbulent boundary layers. It was clearly a three-dimensional problem, probably with relatively thick boundary layers and strong pressure-driven secondary flows due to the S-shaped centerline curvature. Streamwise flow separation was a possibility, at least for the baseline configuration without vortex generators. However, it wasn't necessary to accurately compute any separated flows, since the mere existence of separation would be enough to invalidate the design. Thus, knowing that it separated would be enough. For the operating points of interest, the flow at the entrance to the inlet was expected to be relatively uniform. The principal elliptic effects would be caused by the effect of the S-shaped curvature on the pressure distribution.

Based on the above information, the *RNS3D* parabolized Navier-Stokes code was chosen for this problem. A Reynolds-averaged Navier-Stokes code could also have computed this flow, of course, but would have been much more expensive and time-consuming to run. One particularly nice feature of the *RNS3D* code that made it appropriate for this problem is the fact that it includes the capability to *model* vortex generators, by adding streamwise vorticity to the flow at the location of the generators. Without this capability, it would have been necessary to include the small vortex generators in the actual geometry, greatly complicating and enlarging the computational grid. A Navier-Stokes code would also probably have been required, to compute the shedding of the vortex from the generator trailing edges.

The original version of the *RNS3D* code was called *PEPSIG*, and was developed in the late 1970s and early 1980s. Since then, additional modifications have been made, and the code has been renamed. Like other spatial marching codes, it neglects the viscous and thermal diffusion terms in the streamwise direction. In addition, special treatment is required for the pressure gradient term in the streamwise momentum equation

[3] Vortex generators are small airfoil-shaped devices mounted on a solid surface, usually in groups containing several pairs, that project up into the boundary layer. A vortex is shed from the trailing edge of each generator, and propagates downstream. This enhances the mixing between the high-velocity core flow and the low-velocity boundary layer flow, and can thus delay or prevent flow separation and lower total pressure distortion levels.

to suppress its elliptic behavior. In *RNS3D*, the pressure p in that equation is written as

$$p(x,y,z) = P(x,y,z) + p\prime(x) + p\prime\prime(y,z)$$

where x is the marching direction. Here $P(x,y,z)$ is a known estimate for the pressure field, from a potential flow solution for example; $p\prime(x)$ is a one-dimensional correction computed during the marching solution using global mass flow conservation as a basis; and $p\prime\prime(y,z)$ is a two-dimensional correction in the cross section, also computed during the marching solution.

The details of the derivation of the equations and the solution procedure are beyond the scope of this chapter. The basic analysis is described by Briley and McDonald (1979); Levy, McDonald, Briley, and Kreskovsky (1980); Levy, Briley, and McDonald (1983); and Briley and McDonald (1984). Several reports and papers have been published presenting results of validation studies and applications using the *PEPSIG/RNS3D* code (Towne, 1981, 1984; Vakili, Wu, Hingst, and Towne, 1984; Towne, Povinelli, Kunik, Muramoto, and Hughes, 1985; Towne and Schum, 1985; Anderson, 1986; Kunik, 1986; Povinelli and Towne, 1986; Tsai and Levy, 1987; Anderson, 1991; Anderson and Gibb, 1992; Anderson, Huang, Paschal, and Cavatorta, 1992; Anderson and Towne, 1993).

5.3.2 Geometry Definition

The *Spirit* inlet shown in Figure 5.1 is an S-shaped duct, with elliptical cross sections perpendicular to the centerline. The cross section shape at a particular streamwise station can thus be described by

$$\left(\frac{x_1}{a}\right)^2 + \left(\frac{x_2}{b}\right)^2 = 1$$

where a and b are the semi-major and semi-minor axes of the elliptical cross section, and x_1 and x_2 are local Cartesian coordinates perpendicular to the centerline.

One of the input options in *RNS3D* is to specify the geometry in terms of polynomials in some marching parameter. For this case, the geometric values to be specified were the Cartesian x and y coordinates of the duct centerline, and the semi-major and semi-minor axes a and b.

The centerline coordinates were supplied as cubic splines, thus

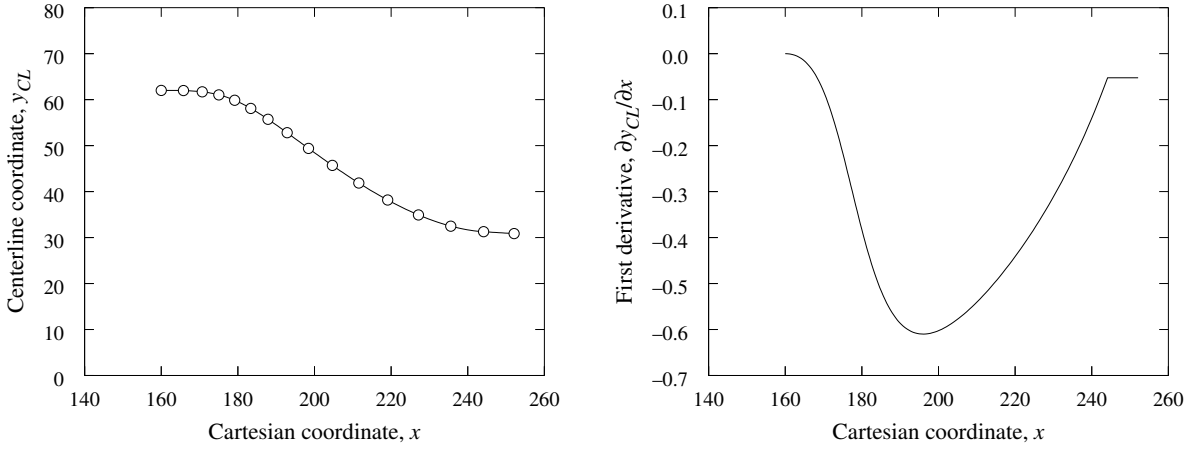
$$x_{CL} = a_0 + a_1 t + a_2 t^2 + a_3 t^3$$
$$y_{CL} = b_0 + b_1 t + b_2 t^2 + b_3 t^3$$

where t was the streamwise marching parameter.

To best fit an actual configuration, *RNS3D* allows the centerline to be split into sections, with different polynomial coefficients in each section. The equations solved by *RNS3D* require second derivatives of the coordinates, which are computed numerically. The geometry description should therefore be smooth from one section to another. In theory, second derivatives of the geometric parameters should be at least continuous. Note that this formally applies to the cross section parameters a and b, as well as the centerline coordinates. Experience with the code, though, shows that in practice getting the centerline smooth is more critical.

The centerline for the *Spirit* inlet was divided into 15 sections. The smoothness was checked by plotting y_{CL} and $\partial y_{CL}/\partial x$ *vs* x, as shown in Figure 5.2. The symbols along the centerline indicate the boundaries between the cubic spline sections defining the shape.

Note the slope discontinuity in the plot of $\partial y_{CL}/\partial x$ near the end of the duct. Initially this caused some concern, but preliminary calculations indicated that it didn't appreciably affect the numerical stability of the CFD calculation, or the computed viscous results, so no additional work was done to eliminate this discontinuity.

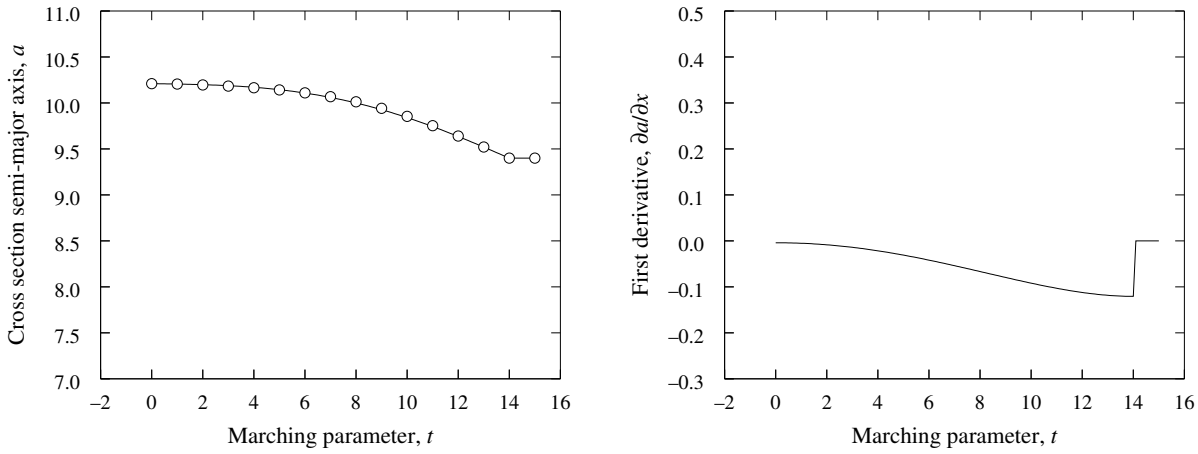


(a) Centerline coordinate, y_{CL}

(b) First derivative, $\partial y_{CL}/\partial x$

Figure 5.2: *Spirit* inlet centerline geometry

The cross section semi-axes a and b were supplied in the form of a table of values *vs* the marching parameter t. A curve fitting routine was used to develop polynomials defining a and b. Like the centerline coordinates, the values of a and b computed from the curve fits were also plotted, both to determine how well the curve fits matched the supplied values, and to examine the smoothness of the result. These plots are shown in Figure 5.3 and Figure 5.4. The symbols in the figures are the supplied tabular values of a and b, and the line is the fitted polynomial.

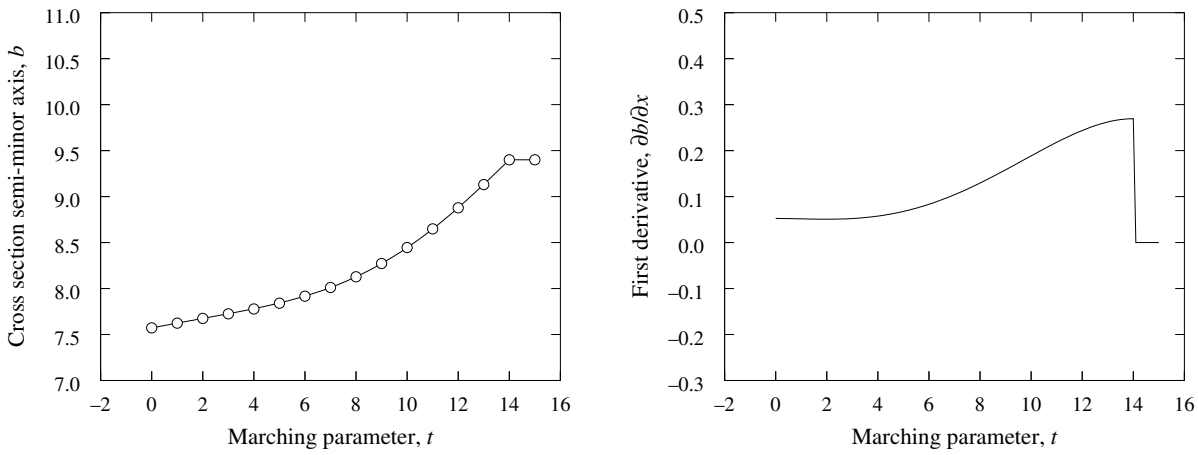


(a) Cross section semi-major axis, a

(b) First derivative, $\partial a/\partial x$

Figure 5.3: Curve fit for semi-major axis

Note that a and b have a discontinuous slope at $t = 14$. This is due to a short, constant-area section at the end of the actual duct. In theory this is probably not a "good thing", but again, preliminary calculations



(a) Cross section semi-minor axis, b

(b) First derivative, $\partial b/\partial x$

Figure 5.4: Curve fit for semi-minor axis

indicated that it didn't adversely affect the computed viscous results.

Besides specifying the geometry in terms of polynomials, another input option in *RNS3D* is to read a 3-D file containing a grid of points defining the surfaces. This option is somewhat more robust and flexible, especially when defining the locations of vortex generators in a duct. The polynomials describing the centerline and cross section axes were thus used to define the boundaries for some preliminary calculations with *RNS3D*. The resulting surface grid was saved in a file, and used as input for subsequent calculations to define the geometry.

5.3.3 Computational Grid

The interior grid point distribution in *RNS3D* is defined at run time via input parameters. Thus, a separate grid generation step was not needed for these calculations. The computational grid used for most of the calculations is shown in Figure 5.5. Note that, since the centerline is two-dimensional and the inlet flow is uniform, the flow will be symmetric about the 0 deg–180 deg line, and only half the cross section needs to be computed.

A 49×49 mesh was used in the 180 deg cross section, with 151 streamwise stations. Grid points were packed in the radial direction near the outer boundary to resolve the boundary layers there. Uniform grid spacing was used in the circumferential and streamwise directions. For clarity, the grid shown in Figure 5.5 has been thinned by a factor of 4 in the radial direction, and 2 in the circumferential and streamwise directions.

5.3.4 Potential Flow Solution

Running a case with *RNS3D* is a two-step procedure. The first step is a potential flow run to compute the pressure estimate $P(x, y, z)$. This potential flow pressure field is saved in a file, and used as input in the second step, the actual viscous marching calculation. Note that the potential flow calculation has to be done only once for a given geometry. Changes in flow conditions, initial profiles, grid density, etc., for the viscous calculation can be made without affecting the pre-stored potential flow pressure file.

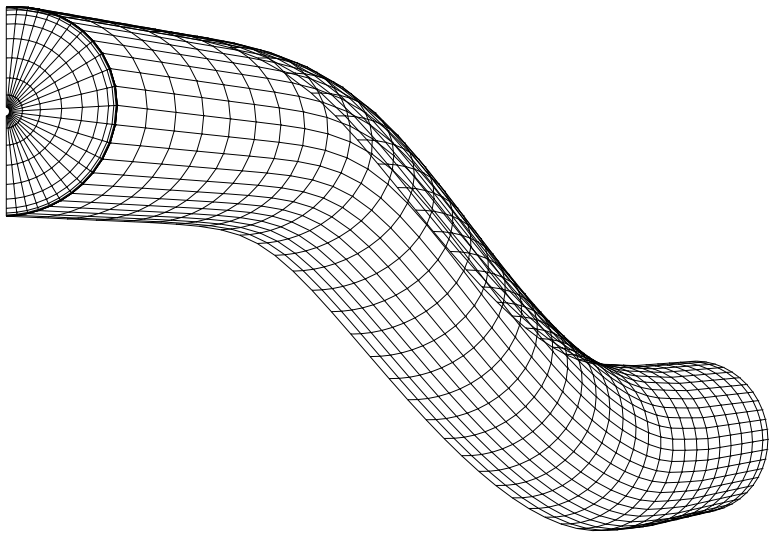


Figure 5.5: Computational grid for the *Spirit* inlet

Since there are no boundary layers to resolve in the potential flow, a coarser mesh may be used. For the *Spirit* inlet, the mesh size was $19 \times 20 \times 75$. The computed potential flow pressure coefficients, along the outer boundary at $\theta = 0$ deg, 90 deg, and 180 deg, and along the centerline, are shown in Figure 5.6.

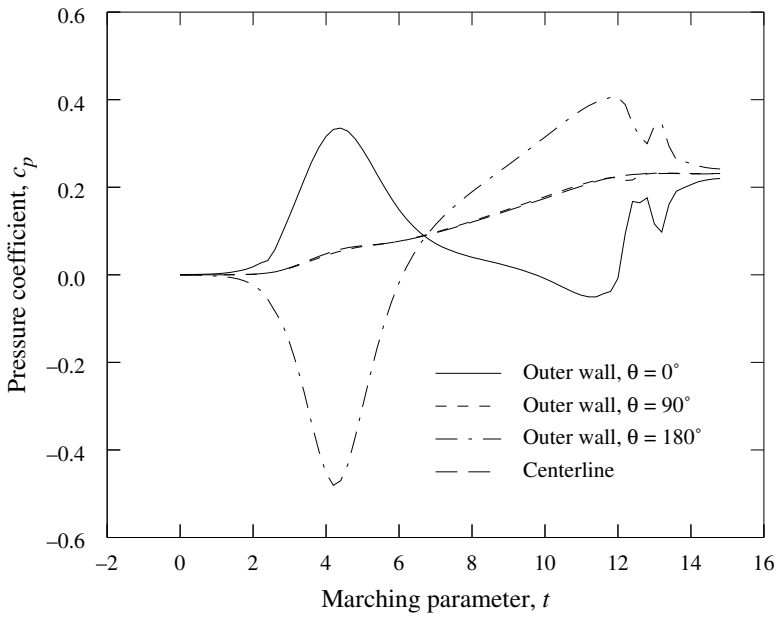


Figure 5.6: Potential flow pressure distribution

The outer wall values at $\theta = 90$ deg and the centerline values are essentially identical, and reflect the increase in cross section area. The outer wall values at $\theta = 0$ deg and 180 deg have the typical shape for an S-duct, with higher pressure on the outside of the bend and lower pressure on the inside. The wiggles at the downstream end are probably due to the discontinuous slope in the cross section axes a and b, described

previously. As noted earlier, they had no significant effect on the viscous solution.

5.3.5 Distortion Criteria

Before describing the viscous calculation, it is useful at this point to discuss the method used to quantify the amount of distortion. The distortion descriptors are based on compressor face total pressure values that normally would be measured in an experiment by a standard 40-probe rake, shown in Figure 5.7. To get analogous results from the CFD calculation, the computed total pressures were interpolated from the much denser computational grid to the probe locations of a 40-probe rake.

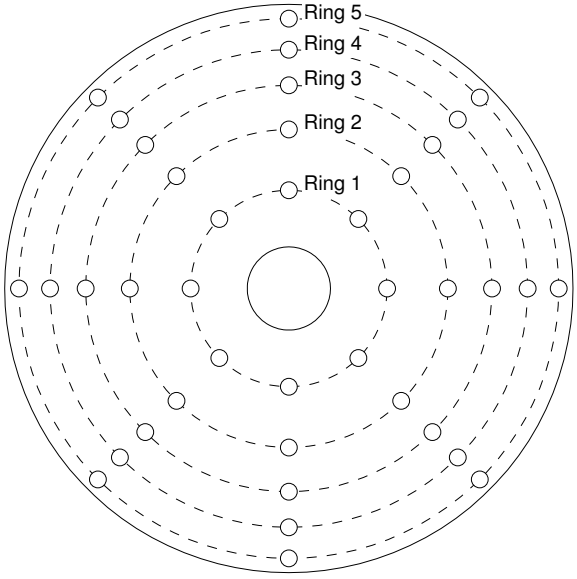


Figure 5.7: Standard 40-probe compressor face rake

For this study, a simplified stability assessment procedure supplied by the engine manufacturer was used. Four distortion descriptors are computed from the compressor face total pressure values, quantifying various aspects of the radial and circumferential distortion. These descriptors are then combined with empirical parameters, which are functions of corrected engine airflow rate, to define DLP(core) and DLP(fan), the distortion limit parameters for the core compressor and the fan tip. Both of these distortion limit parameters must be below 1.0 for stable engine operation.

5.3.6 Viscous Flow Solution/Analysis of Results

The next step was to compute the viscous flow in the inlet without vortex generators. This was done for all three operating conditions listed in Table 5.1. In addition, since no experimental data was available to determine the boundary layer thickness at the throat (the initial station in the marching analysis), each operating condition was run with three different initial boundary layer thicknesses — $\delta/R = 0.025$, 0.05, and 0.10. The development of the flow through the inlet is illustrated in Figure 5.8, in the form of computed total pressure contours at selected streamwise stations, for the high-speed cruise condition with $\delta/R = 0.05$. The horseshoe-shaped pattern at the exit is typical of S-duct flows. The curved centerline causes transverse pressure gradients to be set up in the cross section, as shown by the potential flow results in Figure 5.6.

Pressure-driven secondary flow vortices appear, which drive the low-energy boundary layer flow to the bottom of the duct.

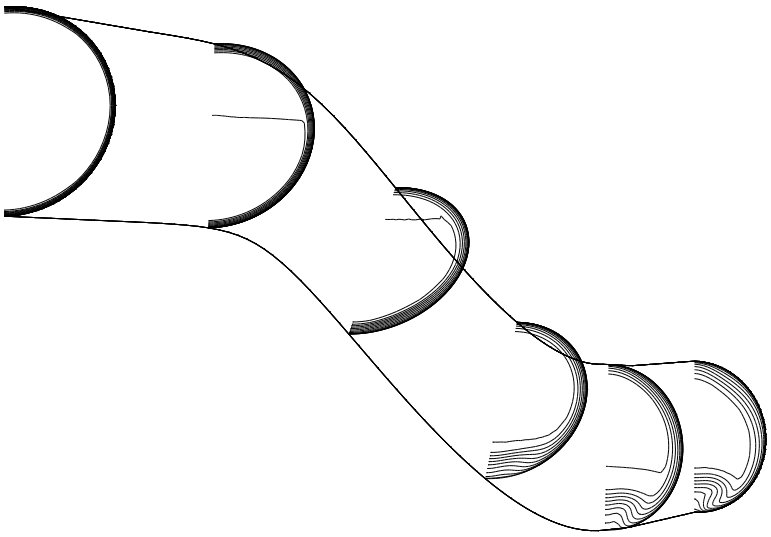


Figure 5.8: Total pressure contours, high-speed cruise condition, without vortex generators

The computed recoveries and distortion limit parameters are shown in Table 5.2 for all three operating conditions and initial boundary layer thicknesses.

Table 5.2: Distortion Limit Parameters Without Vortex Generators

Operating Condition	Initial δ/R	Recovery	DLP(fan)	DLP(core)
High-speed cruise	0.025	0.982	2.156	0.008
	0.05	0.979	2.510	0.076
	0.10	0.972	2.300	0.307
Best cruise	0.025	0.985	0.238	0.024
	0.05	0.982	0.259	0.105
	0.10	0.978	0.264	0.279
Takeoff	0.025	0.992	0.087	0.002
	0.05	0.990	0.107	0.003
	0.10	0.988	0.137	0.037

The total pressure recovery is satisfactory for all operating conditions and inlet boundary layer thicknesses. In addition, both DLP(fan) and DLP(core), the fan tip and core compressor distortion limit parameters, are well below the critical value of 1.0 at the best cruise and takeoff conditions, as is DLP(core) at the high-speed cruise condition. But, DLP(fan) at the high-speed cruise condition is clearly too high. This operating point was thus used to design the vortex generator system for the inlet.

Based on earlier experience in the use of vortex generators in subsonic inlets (Anderson and Gibb, 1992; Anderson, Huang, Paschal, and Cavatorta, 1992), a system was designed for the *Spirit* inlet with eleven pairs

of counter-rotating generators distributed around the 360 deg cross section a short distance downstream of the throat, as shown schematically in Figure 5.9.

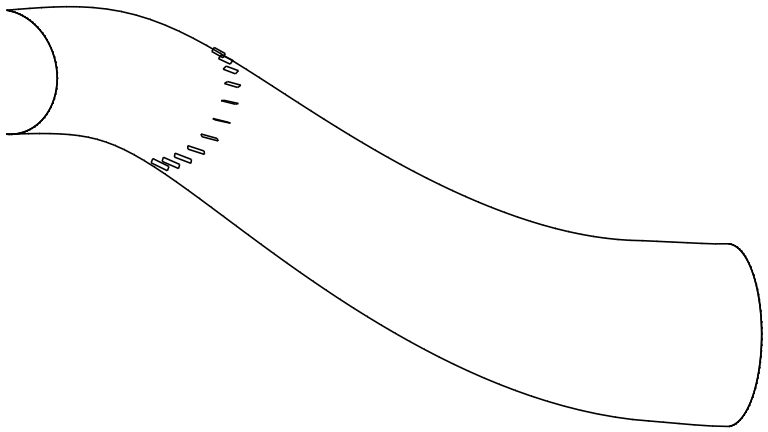


Figure 5.9: Vortex generator installation in the *Spirit* inlet

The design variable that was examined was the generator height h. Cases were run using *RNS3D* with $h/R = 0.04$ to 0.07 in increments of 0.005 for all three initial boundary layer thicknesses. The resulting values for the fan tip distortion limit parameter are shown in Figure 5.10. Based on these results a generator height

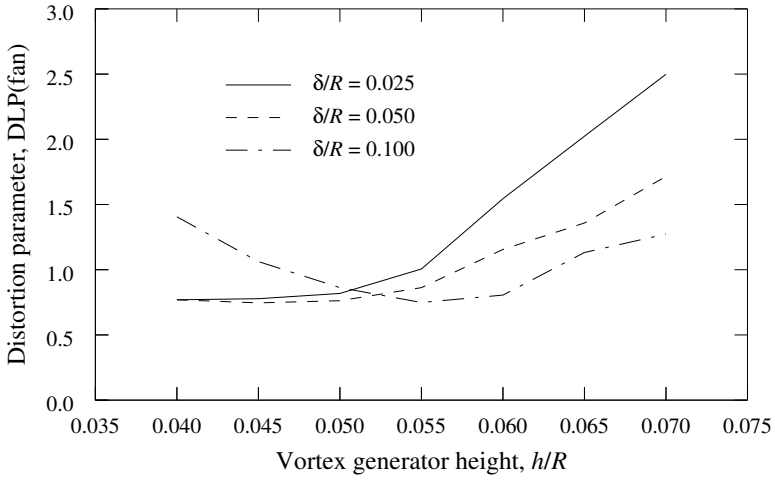


Figure 5.10: Effect of generator height on fan tip distortion limit parameter

of $h/R = 0.05$ was selected as the optimum choice, given the uncertainty in the actual throat boundary layer thickness.

The computed total pressure contours for the $h/R = 0.05$ and $\delta/R = 0.05$ combination are shown in Figure 5.11. By comparing with the results shown in Figure 5.8, it can be seen that the effect of the vortex generators in this case is to split the large region of low total pressure at the compressor face into two smaller regions, and to shift them slightly in the circumferential direction.

Additional computations were also performed to confirm that the distortion levels at the other two operating conditions were still acceptable with vortex generators installed. The results for the three operating

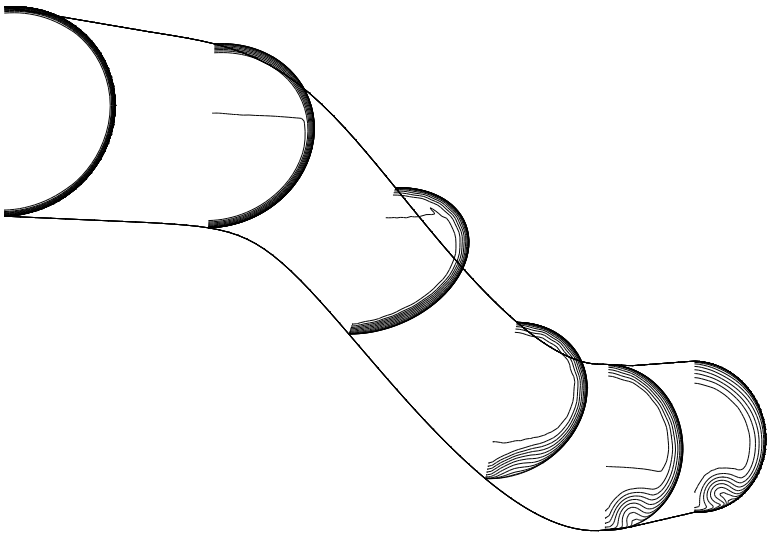


Figure 5.11: Total pressure contours, high-speed cruise condition, with vortex generators

conditions and initial boundary layer thicknesses are summarized in Table 5.3.

Table 5.3: Distortion Limit Parameters With Vortex Generators

Operating Condition	Initial δ/R	Recovery	DLP(fan)	DLP(core)
High-speed cruise	0.025	0.984	0.818	0.008
	0.05	0.982	0.762	0.007
	0.10	0.975	0.862	0.026
Best cruise	0.025	0.989	0.124	0.006
	0.05	0.986	0.144	0.007
	0.10	0.981	0.189	0.038
Takeoff	0.025	0.991	0.095	0.003
	0.05	0.991	0.102	0.004
	0.10	0.988	0.117	0.006

By comparing with the values listed in Table 5.2, it can be seen that all the distortion levels were lowered by the use of vortex generators. The values for DLP(fan) are all below 1.0, the limit set for the candidate engine, but they are still uncomfortably high at the high-speed cruise operating point. The total pressure contours at the compressor face for this condition (see the exit station in Figure 5.11) are actually very similar to the results for the best cruise condition (not shown). The high DLP(fan) values are a result of the large corrected weight flow value of 75.3 lb_m/sec (34.1 kg/sec) at the high-speed cruise condition (see Table 5.1). In the stability assessment procedure used in this study, the DLP(fan) values increase rapidly when the corrected engine airflow increases above 68 lb_m/sec (30.8 kg/sec).

After discussing these results with the inlet designers and the engine manufacturer, it was determined that a lower throat Mach number of 0.61 should have been used for these calculations. At this Mach number, the corrected weight flow is 71.8 lb_m/sec (32.6 kg/sec). The high-speed cruise cases were therefore re-run

with the lower throat Mach number, both with and without vortex generators, and the resulting performance parameters are listed in Table 5.4.

Table 5.4: Distortion Limit Parameters, High-Speed Cruise, $M_{\textit{titthr}} = 0.61$

	Initial δ/R	Recovery	DLP(fan)	DLP(core)
Without vgs	0.025	0.985	0.448	0.007
	0.05	0.981	0.541	0.052
	0.10	0.976	0.564	0.218
With vgs	0.025	0.987	0.259	0.007
	0.05	0.985	0.271	0.006
	0.10	0.979	0.301	0.045

At the lower throat Mach number, the distortion levels are below 1.0 even without vortex generators, although a conservative designer may feel they're still too high. With vortex generators, though, the levels are well below 1.0.

Prior experience with *RNS3D* has shown that having a sufficiently dense mesh, especially in the streamwise direction, is required to get quantitatively accurate predictions of the development of secondary flow vortices. Some additional runs were therefore made for the high-speed cruise condition with vortex generators to investigate the effects of streamwise mesh density. One run was also made doubling the mesh in both cross-flow directions. The results are listed in Table 5.5. The initial boundary layer thickness δ/R for these cases was 0.05.

Table 5.5: Effect of Mesh Density on Distortion Limit Parameters

Mesh	Recovery	DLP(fan)	DLP(core)
$49 \times 49 \times 151$	0.985	0.271	0.006
$49 \times 49 \times 301$	0.982	0.344	0.004
$49 \times 49 \times 601$	0.979	0.411	0.004
$49 \times 49 \times 1201$	0.980	0.379	0.004
$49 \times 49 \times 2401$	0.981	0.363	0.004
$97 \times 97 \times 151$	0.983	0.310	0.023

Using four times as many cross section points increased the computed value of DLP(fan) only slightly. Going from 151 to 601 streamwise points, though, increased the value by over 50%. As the number of points was increased further, DLP(fan) dropped slightly, and appeared to asymptotically approach a value of about 0.35, as shown more clearly in Figure 5.12. This value is still well below the limit of 1.0 for the candidate engine.

Based on these CFD results, Paragon, in consultation with the engine manufacturer, concluded that the *Spirit* inlet would meet the performance criteria for the candidate engine, and that an experimental test

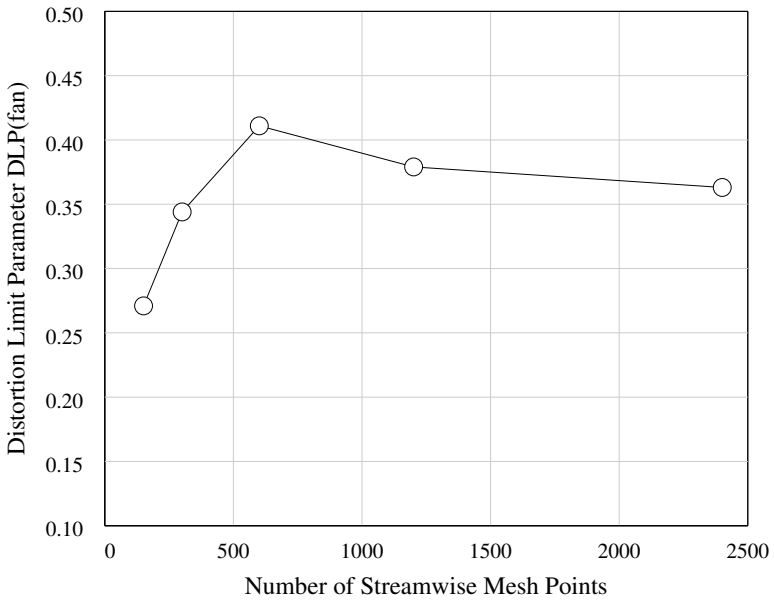


Figure 5.12: Effect of streamwise mesh density on DLP(fan)

program that had been planned could be eliminated with minimal risk. They plan to proceed directly to a flight test with an instrumented inlet installed on the new aircraft. Thus, the careful use of CFD in this project has resulted in significant savings in both cost and time.

5.4 Example — Strut-Jet Engine

Propulsion systems for missiles, reconnaissance aircraft, and single-stage-to-orbit vehicles must operate efficiently at flight conditions ranging from takeoff to hypersonic cruise. Because a specific propulsion cycle is more efficient at one flight condition than others, a new family of combined cycle engines is being studied. These engines combine two or more different propulsion cycles into an integrated system for better overall performance throughout the flight envelope.

One such system currently being studied at the NASA Lewis Research Center is the strut-jet. This engine is based on the Rocket Based Combined Cycle (RBCC) concepts of Escher, Hyde, and Anderson (1995). It combines a high-specific-impulse low-thrust-to-weight airbreathing engine with a low-specific-impulse high-thrust-to-weight rocket engine. From takeoff to high supersonic speeds (about Mach 3) the system operates as an air-augmented rocket. At approximately Mach 3 the rockets are shut down, and the system becomes a dual-mode ramjet. At very high Mach numbers (above about Mach 8) and high altitude, the airbreathing system may not provide adequate thrust, and the rockets would then be turned back on.

Demonstration tests of the strut-jet engine are scheduled to begin at NASA Lewis in the summer of 1996. Thrust measurements will be made at several fuel flow conditions. The measured experimental thrust, however, will include aerodynamic forces on various pieces of attached external hardware, such as the instrumentation and model support system, and therefore will not represent the true thrust of the propulsion system. The true thrust may thus be written as

$$T_{sys} = T_{exp} - T_{ext}$$

where T_{sys} is the true internal thrust of the propulsion system, T_{exp} is the measured thrust in the experiment, and T_{ext} is the (negative) thrust due to external hardware. If the same experiment is run without fuel flow

to the engine, we get

$$(T_{sys})_{nf} = (T_{exp})_{nf} - T_{ext}$$

where $(T_{sys})_{nf}$ is the internal force on the propulsion system, and $(T_{exp})_{nf}$ is the measured force in the experiment. Subtracting, we can write

$$T_{sys} = (T_{sys})_{nf} - \Delta T$$

where $\Delta T = T_{exp} - (T_{exp})_{nf}$ is the measured increment in thrust for a given fuel flow condition. The internal force $(T_{sys})_{nf}$ cannot be measured experimentally because the force T_{ext} due to external hardware cannot be determined independently. $(T_{sys})_{nf}$ can be computed using CFD, however, allowing the true thrust of the propulsion system to be determined.

To compute this internal force, the NPARC code will be used. NPARC is a multi-block Navier-Stokes code being developed and supported by the NPARC Alliance, a partnership between the NASA Lewis Research Center and the USAF Arnold Engineering Development Center (NPARC Alliance, 1994). It solves the Reynolds-averaged, unsteady compressible Navier-Stokes equations in generalized nonorthogonal body-fitted coordinates. Several turbulence models are available in the code; for this application, the Chien low Reynolds number k-ϵ model will be used (Chien, 1982). Spatial derivatives in NPARC are represented using central difference formulas, and explicit boundary conditions are used. Jameson's artificial dissipation model is used for stability, and to smooth pre- and post-shock oscillations and to prevent odd-even point decoupling (Jameson, Schmidt, and Turkel, 1981). The equations are solved by marching in time using an ADI algorithm derived using the Beam-Warming approximate factorization scheme (Beam and Warming, 1978).

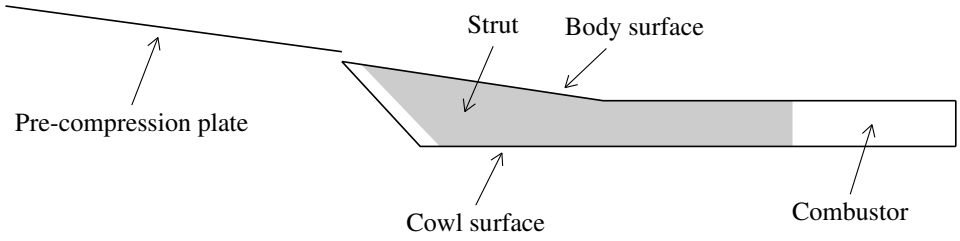
After the NPARC CFD calculations have been completed, the solution will be post-processed to obtain the internal force $(T_{sys})_{nf}$. Two calculation methods will be used. The first is a simple momentum balance, subtracting the integrated momentum at the duct entrance from the integrated momentum at the exit. The momentum is computed by numerical integration over the computational grid. The second method integrates the pressure and skin friction forces on the internal surfaces of the configuration to obtain the internal force. Ideally, these two methods will give identical results. However, several sources of error can contribute to a discrepency. Incomplete mass continuity and difficulty in calculating accurate skin friction are the two most common problems.

This methodology has been tested on a subscale model of the strut-jet engine that was tested in the NASA Lewis 1×1 ft (0.3×0.3 m) supersonic wind tunnel (Fernandez, Trefny, Thomas, and Bulman, 1996). A simplified schematic of the model is shown in Figure 5.13.

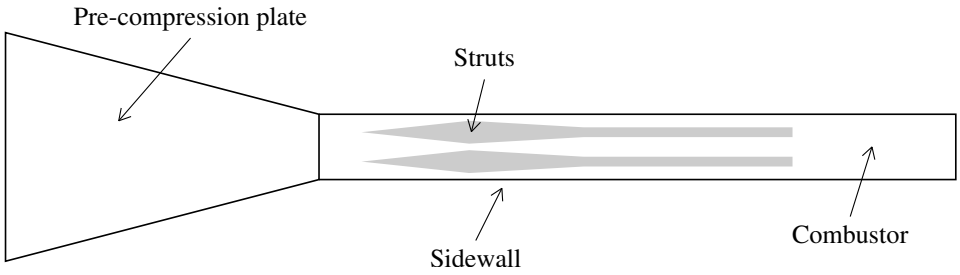
The strut-jet configuration that was analyzed consisted of a rectangular cross-sectioned inlet with swept leading-edge sidewalls. Two struts, also with swept leading edges, were installed in the inlet. In the actual engine test that will be run in the Hypersonic Test Facility, the rockets will be installed in the base of these struts. A pre-compression plate upstream of the inlet was used to simulate the effect of the vehicle forebody.

The computational grid was created using GRIDGEN, a grid generation package widely used for CFD applications (Steinbrenner, Chawner, and Fouts, 1990). Six grid blocks were used, as listed in Table 5.6, with a total of 1,400,319 points. Note that the configuration is symmetric, and thus only half the duct was computed, from the sidewall to the center symmetry plane between the two struts.

For these calculations, the free stream Mach number, static pressure, and static temperature were 6.0, 14.98 $\mathrm{lb_f/ft^2}$ (717.2 $\mathrm{N/m^2}$), and 93.13 °R (51.74 K), respectively. The resulting Reynolds number was 3.80×10^6/ft (12.5×10^6/m). Convergence was achieved after the L_2 norm of the residual was reduced at least three orders of magnitude in each grid block, and when no discernable change in the static pressure and mass flow distributions were observed over at least 1,000 iterations. Representative results are shown in Figure 5.14, where the computed and experimental static pressures distributions are plotted along the body and cowl centerlines.

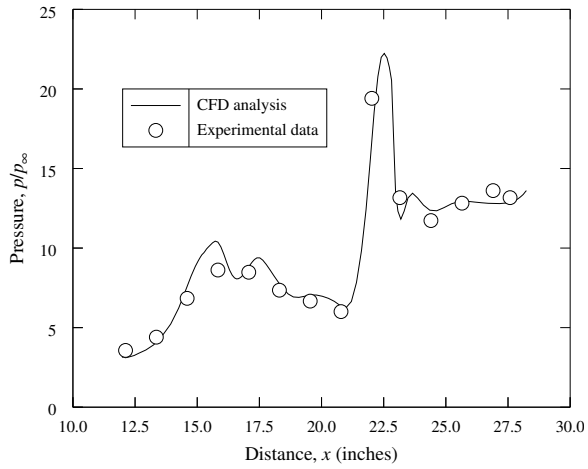


(a) Sidewall removed

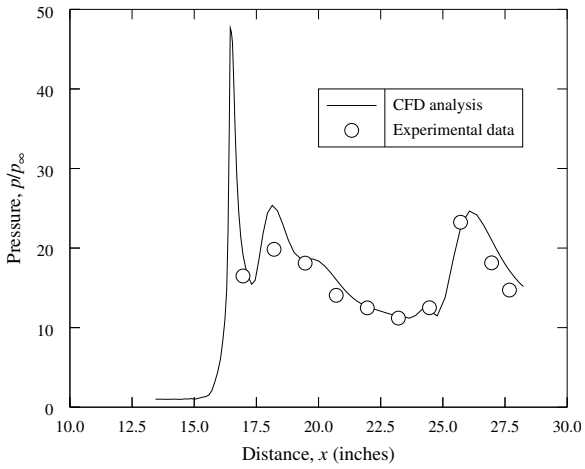


(b) Cowl removed

Figure 5.13: Rocket based combined cycle engine



(a) Along body centerline



(b) Along body cowl

Figure 5.14: Pressure distribution in strut-jet engine

The internal force values computed using the two calculation methods were within 1.5% of each other, as shown in Table 5.7. Positive values indicate thrust, and negative values indicate drag.

This example is an illustration of how CFD can be used to solve a problem that would be very difficult and

Table 5.6: Grid Blocks for Strut-Jet Engine Computation

Block No.	Grid Size	Description
1	$22 \times 76 \times 30$	Inlet entrance center duct
2	$111 \times 57 \times 30$	Center duct
3	$111 \times 57 \times 51$	Side duct
4	$22 \times 76 \times 52$	Inlet entrance side duct
5	$59 \times 95 \times 80$	Forebody
6	$51 \times 57 \times 104$	Combustor section

Table 5.7: Internal Force Balance for Strut-Jet Engine

Momentum Balance		Force Integration		
Boundary	**Momentum, $\mathbf{lb_f}$ (N)**	**Surface**	**Pressure, $\mathbf{lb_f}$ (N)**	**Skin Friction, $\mathbf{lb_f}$ (N)**
Inflow	−32.3999 (−144.122)	Body	−1.3563 (−6.033)	−0.4740 (−2.108)
Spillage	0.8108 (3.607)	Cowl	0.0000	−0.3226 (−1.435)
Outflow	28.3773 (126.228)	Strut	0.4575 (2.035)	−1.1703 (−5.206)
		Sidewall	0.0000	−0.6499 (−2.891)
		Base	0.2566 (1.141)	0.0000
Total	−3.2118 (−14.287)	Total	−3.2590 (−14.497)	

expensive to solve any other way. Experimentally, there is no practical way to separate the true propulsion system thrust from the measured thrust at these hypersonic conditions. Using CFD, however, allows the true thrust of the propulsion system to be determined.

5.5 Current Status and Future Directions

Over the last 10 years or so, several papers and journal articles have appeared describing the status of CFD for applications. By their very nature, of course, publications like these become outdated fairly quickly. This section presents this author's perception of the current status of CFD for inlet, duct, and nozzle applications, and indicates possible future directions that would make CFD more useful in the real world (i.e., industry). Much of the material here has been influenced by the authors of two recent papers on the use of CFD in the aerospace industry (Cosner, 1994; Paynter, 1994).

There are several issues, sometimes overlapping, that are inhibiting the widespread routine use of CFD, especially in the design environment. Some of these are modeling issues, resulting from our lack of understanding of some of the basic but complex flow physics in many real-world applications. Some are numerical issues, dealing with how the equations are solved. Others are more procedural in nature, related to how the various steps involved in a CFD analysis are currently being accomplished, and to how CFD codes are written, tested, and evaluated.

5.5.1 Modeling Issues

For non-reacting flow through inlets, ducts, and nozzles, the principal flow modeling problems remaining today are:

- *Turbulence*

  As noted earlier, a universal turbulence model that works well for all types of flow does not yet exist. In general, guidelines based on experience must be used when choosing the model to use for a particular problem. This is an active area of research, and new turbulence models, or variations on existing ones, seem to be proposed weekly. This rapid growth makes it difficult to evaluate new models, however. The situation would improve with the development and acceptance of standards for software interfaces, validation, and documentation. (See below.)

- *Laminar-turbulent transition*

  Our capability to predict laminar-turbulent transition is even less mature than that for fully-turbulent flows. In many CFD codes the flow must be either fully laminar or fully turbulent. Those that are capable of predicting transition generally use fairly crude models based on correlations with experimental data for simple flows. Better models are needed for use in computing realistic 3-D flows in engineering applications. See the recent papers by Simon (1993) and Simon and Ashpis (1996) for an overview of the current research in this area.

- *Boundary conditions*

  CFD is typically used to study the flow through a component of a larger physical system, such as the inlet in a jet engine. At some types of boundaries, such as a simple no-slip solid wall or a supersonic inflow boundary, choosing appropriate boundary conditions is fairly straightforward. For many other types of boundaries, the situation is more complicated. The conditions specified at the outflow boundary of an inlet, for example, must properly model the influence of the compressor on the flow in the inlet. Porous wall boundary conditions are normally used to represent the flow through a bleed region in a supersonic inlet. These specialized boundary conditions must also be able to model unsteady

interactions between components, such as the reflection at the compressor face of a disturbance in the inlet flow. Additional research is required to develop satisfactory boundary conditions for specialized applications like these.

5.5.2 Numerical Issues

The numerical algorithms being used in modern CFD codes to solve the governing equations are generally pretty fast. While faster algorithms are always desirable, other numerical issues are also of critical importance.

- *Computational platform*

  The computational power available to the CFD user has increased dramatically over the last 10–15 years. In the not-too-distant past, CFD codes were almost always run on large mainframe computers, but today Navier-Stokes analyses for relatively simple 2-D problems can be run on desktop PCs and workstations. Even some 3-D problems are being run on mid-range to high-end workstations. Parallel processing software has been developed that allows CFD codes to use multiple processors, either on a single computer with multiple CPUs or on a cluster of computers, with each processor computing a part of the problem.

  This rapid and continual growth in the capability of the hardware has in many respects been the determining factor in the growth of CFD. The computer speed and memory that is available to the CFD user influences, for example, the size of the grid and the sophistication of the turbulence model. As the hardware continues to improve, CFD simulations will also continue to improve.

  There are other advancements that are possible in CFD, besides those related to the raw speed and memory of the computer. New solution algorithms that are designed specifically to take advantage of parallel processing capabilities should be investigated. Faster algorithms may also be developed by taking advantage of other features present in a specific type of computational architecture. The disadvantage to this, and it's a big one, may be lack of portability between platforms. For long-term use in a design environment, the trade-off is probably not worthwhile.

- *Unsteady flows*

  While many CFD codes are at least theoretically able to compute unsteady flows, not much emphasis has been placed on their numerical accuracy. Interest in unsteady flows is growing, however, and this area will see increasing activity in the future.

5.5.3 Procedural Issues

Another reason, perhaps the main reason, that CFD is not more widely used in routine design work is that the process is still too difficult and time-consuming, especially for non-CFD experts. There are several, sometimes inter-related, factors involved.

- *Ease of use*

  The computer codes used in CFD, from the pre-processors used to define the geometry and generate the grid, through the post-processors used to analyze the results, need to be made simpler to use. Until fairly recently CFD was basically a research area, with much effort being put into the development of faster and more accurate solution algorithms. Ease of use for the non-expert user was not a high priority for the code developer. However, the situation is changing. Solution algorithms are now pretty good, as noted above, and ease of use is becoming more important.

Part of the solution will require closer coupling between the various steps in the CFD process, and the development of various standards will help, as discussed below. There are other things that should also be done, however, to make the individual steps in the solution process easier.

For complex configurations, grid generation is currently one of the more time-consuming steps. In particular, setting up the various blocks for a multi-block analysis and linking the grid blocks together can be very labor intensive (Cosner, 1994). Automating this step as much as possible would be very beneficial. Cosner suggests an expert system type of approach, in which the key geometric features would be identified, and, along with the expected flow conditions, used as input to a system that would recommend the layout of the grid blocks, plus the grid size and grid point distribution within the blocks. A corollary to this idea is the use of adaptive grid techniques, in which the grid points are automatically redistributed to resolve high-gradient regions as the flow is being computed. Adaptive gridding is not a new idea, and has already been demonstrated for a variety of problems. However, it has not yet become a standard feature in most CFD analysis systems, perhaps because it requires close coupling between the grid generator and the flow solver. Adaptive gridding has the potential, though, to make the initial grid generation step much easier, to eliminate much of the manual iteration that is now sometimes necessary between the grid generator and flow solver, and to allow the best possible solution for a given number of grid points.

The flow solvers themselves can also be made easier to use. Some CFD codes are overly sensitive to things like nonorthogonal and non-smooth meshes, the time step size, and the choice of artificial viscosity parameters. Getting good results (or in the most extreme cases, any results at all) from a CFD code may require "tweaking" the input until the "correct" value or combination of values is found. More robust solution algorithms, that are less sensitive to mesh and input irregularities, would help. Another improvement would be the development of an intelligent user interface, that could be used to help set up the input for a particular problem and to check it for inconsistencies.

Post-processing systems, while generally very good, can still be improved. As noted earlier, with the interactive 3-D graphics packages available today, results may be displayed and examined in almost any form imaginable. While this can be tremendously useful, for the most part the user must visually examine the computed results. More automated methods should be developed to identify key flow features and problem areas. The capability to perform solution quality checks, similar to the grid quality checks already available in some grid generation codes, is also needed. These automated post-processing capabilities, once available, should be used to examine the flow field as it is being computed, and recommend changes to the grid and/or input parameters where appropriate.

- *Standardization*

  The various steps in the solution process have historically been separate elements. As a result, too much of the time required to solve a problem is spent in between the elements, converting the output from one step to the input for the next step. In addition, there is often too much iteration required between steps (e.g., "change the grid and recompute"). Closer coupling between the various steps is needed, so that the entire solution process from the geometry specification through the analysis of the results becomes as seamless as possible. Using CFD for design requires, almost by definition, the ability to easily change the geometry and determine the effect of that change on the flow.

  To accomplish this, standards need to be developed and accepted by the CFD community for the interfaces between the various steps. For example, the wide variety of CAD packages in use for geometry specification have resulted in a variety of formats for the CAD output, many of them not directly readable by popular grid generation programs. CFD flow solvers read grid files in a variety of formats, and there is no universal standard for the interface requirements and boundary conditions to be used between blocks in a multi-block grid. The *PLOT3D* format (Walatka, Buning, Pierce, and Elson, 1990) has become a de facto standard for the output from CFD codes, and can be read by a variety of post-processors. Unfortunately, this format does not include all the information necessary to fully represent some computed results, such as turbulence data.

A variety of more complete formats have been or are being developed, such as the NASA-IGES standard for CAD output (Blake, Kerr, Thorp, and Chou, 1991), and the interface standards from the NASA-funded Complex Geometry Navier-Stokes (CGNS) project. Since there is no "governing body" in CFD, however, the development of a single accepted standard for the interface between each step in the solution process is unlikely, at least in the near future. Instead, several "standards" will probably co-exist. Code developers should therefore strive to support directly as wide a variety of the proposed formats as possible, both for input and output. In addition, generalized interface routines should be developed to convert data between a variety of standard formats.

Besides the need for standard data formats between steps in the solution process, standard interfaces are needed between modules within the individual codes. This is especially true for the CFD flow solver itself, where standard subprogram interfaces would make the development and testing of new technology, such as improved turbulence models, much easier. The CGNS project is addressing this issue also.

And finally, research into multi-disciplinary methods, such as a CFD analysis coupled with an elastic structure analysis, is increasing. For these methods to ultimately be useful in the real world, standards are required for sharing data between the multiple analyses involved.

- *Validation*

  CFD code validation has been the subject of much interest in recent years (e.g., Marvin, 1993; Mehta, 1995; Aeschliman, Oberkampf, and Blottner, 1995). While various terms, such as verification, certification, and validation, have been used to describe different aspects of the process, it basically refers to determining how well a CFD code is able to simulate reality. In order to determine the strengths and weaknesses of a CFD code, cases should be run for a variety of geometric configurations, and over a range of flow parameters. Computed results should be compared with benchmark-quality experimental data, well-accepted computational results, and/or analytic solutions. If CFD is to become an accepted tool for design, code validation must be emphasized. The code developer must demonstrate that his/her code is able to simulate reality accurately enough, and quickly enough, to be relied upon in a design environment.

  Starting as far back as 1968, various organizations have developed databases containing standard sets of experimental data to be used for CFD validation for various types of flow (e.g., Coles and Hirst, 1968; AGARD, 1988; Settles and Dodson, 1991). As our capability to predict more complex flows increases, the need for high-quality experimental validation data for those flows also increases. Data are now needed for high (i.e., flight level) Reynolds number turbulent flows, low Reynolds number transitional flows, and unsteady flows. These data would be especially useful for evaluating newly-proposed turbulence models.

- *Documentation*

  CFD codes are notoriously poorly documented. For many CFD codes the documentation, if it exists at all, consists only of a User's Guide describing the input and output, with a few examples. It generally does not include a detailed description of the code itself, showing exactly how the various physical and numerical models involved have been implemented. These details are often not even described in comments within the code itself. Without this information, even a knowledgeable CFD researcher will have difficulty modifying the code to test hypotheses about the cause of any disagreement with experimental data in a validation study.

  Papers presenting applications of CFD are also often poorly documented. In addition to describing the problem and the CFD method that was used, they should include at least brief descriptions of the turbulence model, any artificial viscosity that was used, the grid size and distribution, the boundary conditions, and, for iterative methods, the iteration method and convergence history. Without these details, it is difficult or impossible to assess the significance of the computed results.

5.6 Acknowledgements

The author would like to acknowledge and thank his co-workers at NASA Lewis Research Center and in the NPARC Alliance for their valuable contributions to this chapter. Special thanks go to Bernie Anderson and Julie Dudek, who did most of the work on the Paragon *Spirit* inlet calculations; to Jim DeBonis and Shaye Yungster, who supplied the material on the strut-jet engine; and to Ray Cosner and Jerry Paynter, who have provided insight into the difficulties involved in using CFD for real-world design in an industrial environment.

References

Aeschliman, D. P., Oberkampf, W. L., and Blottner, F. G. (1995) "A Proposed Methodology for Computational Fluid Dynamics Code Verification, Calibration, and Validation," 16th. International Congress on Instrumentation in Aerospace Simulation Facilities, Wright-Patterson AFB, OH, Jul 18–21, 1995.

AGARD (1988) "Validation of Computational Fluid Dynamics, Volume 1 — Symposium Papers and Round Table Discussion," AGARD-CP-437.

Ames Research Staff (1953) "Equations, Tables, and Charts for Compressible Flow," NACA Report 1135.

Anderson, J. D. (1982) *Modern Compressible Flow With Historical Perspective*, McGraw-Hill Book Company.

Anderson, B. H. (1986) "Three-Dimensional Viscous Design Methodology of Supersonic Inlet Systems for Advanced Technology Aircraft," in *Numerical Methods for Engine-Airframe Integration*, Murthy, S. N. B., and Paynter, G. C., eds., Progress in Astronautics and Aeronautics series, AIAA, New York, 1986.

Anderson, B. H. (1991) "The Aerodynamic Characteristics of Vortex Ingestion for the F/A-18 Inlet Duct," AIAA Paper 91-0130.

Anderson, B. H., and Gibb, J. (1992) "Application of Computational Fluid Dynamics to the Study of Flow Control for the Management of Inlet Distortion," AIAA Paper 92-3177.

Anderson, B. H., Huang, P. S., Paschal, W. A., and Cavatorta, E. (1992) "A Study on Vortex Flow Control of Inlet Distortion in the Re-Engined 727-100 Centre Inlet Duct Using Computational Fluid Dynamics," AIAA Paper 92-0152.

Anderson, B. H., and Towne, C. E. (1993) "Application of Computational Fluid Dynamics to Inlets," in *Practical Intake Aerodynamic Design*, Goldsmith, E. L., and Seddon, J., eds., Blackwell Scientific Publications, Oxford, and AIAA, New York, 1993.

Beam, R. M., and Warming, R. F. (1978) "An Implicit Factored Scheme for the Compressible Navier-Stokes Equations," AIAA Journal, Vol. 16, No. 4, pp. 393–402.

Blake, M. W., Kerr, P. A., Thorp, S. A., and Chou, J. J. (1991) "NASA Geometry Data Exchange Specification for Computational Fluid Dynamics," NASA RP 1338.

Briley, W. R., and McDonald, H. (1979) "Analysis and Computation of Viscous Subsonic Primary and Secondary Flows," AIAA Paper 79-1453.

Briley, W. R., and McDonald, H. (1984) "Three-Dimensional Viscous Flows With Large Secondary Velocity," J. Fluid Mech., Vol. 144, pp. 47–77, 1984.

Chien, K.-Y. (1982) "Predictions of Channel and Boundary-Layer Flows with a Low-Reynolds-Number Turbulence Model," AIAA Journal, Vol. 20, No. 1, pp. 33–38.

Coles, D. E., and Hirst, E. A. (1968) "Computation of Turbulent Boundary Layers — 1968 AFOSR-IFP-Stanford Conference, Volume I and II," Stanford University.

Cosner, R. R. (1994) "Issues in Aerospece Application of CFD Analysis," AIAA Paper 94-0464.

Escher, W. J. D., Hyde, E. H., and Anderson, D. M. (1995) "A User's Primer for Comparative Assessments of All-Rocket and Rocket-Based Combined-Cycle Propulsion Systems for Advanced Earth-to-Orbit Space Transport Applications," AIAA Paper 95-2474.

Fernandez, R., Trefny, C. J., Thomas, S. R., and Bulman, M. (1996) "Parametric Data from a Wind Tunnel Test on a Rocket Based Combined Cycle Engine Inlet," NASA TM 107181.

Jameson, A., Schmidt, W., and Turkel, E. (1981) "Numerical Solutions of the Euler Equations by Finite Volume Methods Using Runge-Kutta Time-Stepping Schemes," AIAA Paper 81-1259.

Kunik, W. G. (1986) "Application of a Computational Model for Vortex Generators in Subsonic Internal Flows," AIAA Paper 86-1458 (also NASA TM 87327).

Levy, R., Briley, W. R., and McDonald, H. (1983) "Viscous Primary/Secondary Flow Analysis for Use with Nonorthogonal Coordinate Systems," AIAA Paper 83-0556.

Levy, R., McDonald, H., Briley, W. R., and Kreskovsky, J. P. (1980) "A Three-Dimensional Turbulent Compressible Subsonic Duct Flow Analysis for Use with Constructed Coordinate Systems," AIAA Paper 80-1398.

Marvin, J. G. (1993) "Dryden Lectureship in Research, A Perspective on CFD Validation," AIAA Paper 93-0002.

Mehta, U. B. (1995) "Guide to Credible Computational Fluid Dynamics Simulations," AIAA Paper 95-2225.

NPARC Alliance (1994) "A User's Guide to NPARC Version 2.0."

Numbers, Keith E. (1994) "Survey of CFD Applications for High Speed Inlets," WL-TR-94-3131.

Paynter, Gerald C. (1994) "CFD Status for Supersonic Inlet Design Support," AIAA Paper 94-0465.

Povinelli, L. A., and Towne, C. E. (1986) "Viscous Analyses for Flow Through Subsonic and Supersonic Intakes," NASA TM 88831 (Prepared for the AGARD Propulsion and Energetics Panel Meeting on Engine Response to Distorted Inflow Conditions, Munich, Germany, Sept. 8–9, 1986).

Settles, G. S., and Dodson, L. J. (1991) "Hypersonic Shock/Boundary-Layer Interaction Database," NASA CR 177577.

Simon, F. F. (1993) "A Research Program for Improving Heat Transfer Prediction for the Laminar to Turbulent Transition Region of Turbine Vanes/Blades," NASA TM 106278.

Simon, F. F. and Ashpis, D. E. (1996) "Progress in Modeling of Laminar to Turbulent Transition on Turbine Vanes and Blades," NASA TM 107180. (Submitted for publication in the International Journal of Heat Transfer and Fluid Flow).

Steinbrenner, J. P., Chawner, J. R., and Fouts, C. R. (1990) "The GRIDGEN 3D Multiple Block Grid Generation System," WRDC-TR-90-3022.

Towne, C. E. (1984) "Computation of Viscous Flow in Curved Ducts and Comparison with Experimental Data," AIAA Paper 84-0531 (also NASA TM 83548).

Towne, C. E., and Anderson, B. H. (1981) "Numerical Simulation of Flows in Curved Diffusers with Cross-Sectional Transitioning Using a Three-Dimensional Viscous Analysis," AIAA Paper 81-0003 (also NASA TM 81672).

Towne, C. E., Povinelli, L. A., Kunik, W. G., Muramoto, K. K., and Hughes, C. E. (1985) "Analytical Modeling of Circuit Aerodynamics in the New NASA Lewis Altitude Wind Tunnel," AIAA Paper 85-0380 (also NASA TM 86912).

Towne, C. E., and Schum, E. F. (1985) "Application of Computational Fluid Dynamics to Complex Inlet Ducts," AIAA Paper 85-1213 (also NASA TM 87060).

Tsai, T., and Levy, R. (1987) "Duct Flows With Swirl," AIAA Paper 87-0247.

Vakili, A., Wu, J. M., Hingst, W. R., and Towne, C. E. (1984) "Comparison of Experimental and Computational Compressible Flow in an S-Duct," AIAA Paper 84-0033.

Walatka, P. P., Buning, P. G., Pierce, L., and Elson, P. A. (1990) "PLOT3D User's Manual," NASA TM 101067.